TEXAS A&M UNIVERSITY
PROJECT THEMIS

Technical Report No. 23

CONVEX: A COMPUTER PROGRAM FOR SOLVING CONVEX PROGRAMS

TEXAS A&M UNIVERSITY

COLLEGE STATION, TEXAS 77843

THEMIS

TEXAS A&M UNIVERSITY
PROJECT THEMIS

Technical Report No. 23

CONVEX: A COMPUTER PROGRAM FOR SOLVING CONVEX PROGRAMS

by

H. O. Hartley, R. R. Hocking, L. R. LaMotte, and H. H. Oxspring

Texas A&M Research Foundation
Office of Naval Research
Contract N00014-68-A-0140
Project NR047-700

ATTACHMENT I

CONVEX: A COMPUTER PROGRAM FOR SOLVING CONVEX PROGRAMS

by

H. O. Hartley, R. R. Hocking, L. R. LaMotte, and H. H. Oxspring

THEMIS OPTIMIZATION RESEARCH PROGRAM
Technical Report No. 23
July 1970

INSTITUTE OF STATISTICS
Texas A&M University

Research conducted through the
Texas A&M Research Foundation
and sponsored by the
Office of Naval Research
Contract N00014-68-A-0140
Project NR047-700

ATTACHMENT II

CONVEX: A COMPUTER PROGRAM FOR SOLVING CONVEX PROGRAMS

by

H. O. Hartley, R. R. Hocking, L. R. LaMotte, H. H. Oxspring

Introduction

The purpose of this technical report is to describe a computer program implementing the convex programming algorithm developed by Hartley and Hocking (1963). The program has been used successfully on a number of problems connected with Project Themis as well as numerous other problems which have arisen at Texas A&M. The program is also being used by a number of other institutions as shown on the attached List of Users.

The report is in two parts. Part I is a description of the algorithm as extracted from the initial paper and Part II gives a documentation of the program.

PART I: Description of the Algorithm

INTRODUCTION

The general mathematical programming problem can be described as

$$\text{maximize} \quad g(x)$$
$$\text{subject to} \quad f_i(x) \leq 0 \quad i=1 \ldots m$$

The convex programming problem is an important special case which is amenable to mathematical and numerical treatment and requires that $g(x)$ is concave and the $f_i(x)$ are convex, real-valued functions of the n-vector X for all real X. The non-negative condition on the variables may be included but it introduces no loss of generality. It is also assumed that the functions are differentiable. Certain special problems exist in which these properties hold only in a subset of the n-space but they can frequently be handled by slight modifications of the algorithm to be discussed in the following.

Formulation of the Convex Problem as a Linear Problem

The problem given above is equivalent to that of maximizing the co-ordinate X_{n+1} subject to the original restrictions and the additional restriction $X_{n+1} - g(x) \leq 0$. Note that the concavity of $g(x)$ implies that this new restriction is convex and could just be adjoined to the original set as $f_{m+1}(x) \leq 0$. It does have particular significance however and so it will be carried in its original form.

This new problem has a convex feasible region in (n+1) - space bounded on the sides by cylindrical sets formed from the boundaries $f_i(x)=0$ and $x=0$ and bounded from above by the surface $x_{n+1} = g(x)$. This feasible region is now approximated by a new one which is defined by hyperplanes in (n+1) - space, that is, the convex feasible region defined by the restrictions $f_i(x) \leq 0 \quad i = 1 \ldots m + 1$ is replaced by one which is defined by the intersection of a large set of linear half-spaces. These hyperplanes are constructed as follows:

(i) Impose an arbitrarily fine grid on n-space.

(ii) Construct a tangent plane to each surface $x_{n+1} = f_i(x)$ $i = 1 \ldots m + 1$ at each grid point.

(iii) For $i = 1 \ldots m$ consider the intersection of this plane with $x_{n+1} = 0$ and use the resulting linear restriction. For $i = m + 1$ use the tangent plane itself as a linear boundary to the feasible space.

The general equation of a tangent plane to the surface $x_{n+1} = f(x)$ at a grid point x^* is given by

$$x_{n+1} = \sum_{j=1}^{n} \left.\frac{\partial f(x)}{\partial x_j}\right]_{x^*} (x_j - x_j^*) + f(x^*)$$

Thus for each of the restrictions $f_i(x) \leq 0 \quad i = 1 \ldots m$ we have a set of linear restrictions of the form

$$\sum_{j=1}^{n} \left.\frac{\partial f_i(x)}{\partial x_j}\right]_{x^*} (x_j - x_j) + f_i(x^*) \leq 0$$

Similarly for the restriction $x_{n+1} - g(x) \leq 0$ we get the set of linear restrictions

$$x_{n+1} - \sum_{j=1}^{n} \left.\frac{\partial g(x)}{\partial x_j}\right]_{x^*} (x_j - x_j^*) - g(x^*) \leq 0$$

At this point the additional assumption is made that the original feasible region be bounded. Thus there is some value D such that the cube $|x_j| \le D$, $j = 1 \ldots n$ contains the feasible region. Thus if we only allow grid points in this cube the linear problem formulated above has a finite number of restrictions and the simplex method could, at least theoretically be applied. (in practice it has not been found necessary to specify D.)

The concavity of $g(x)$ on a bounded space allows us to postulate a value M such that the optimum value of $g(x) \le M$.

The linear problem formulated above can now be summarized as follows:

$$\max d'x$$

s.t.

$$A x \le c$$

$$x_i \ge 0 \quad i = 1 \ldots n$$

$$x_{n+1} \text{ unrestrained}$$

Here $x' = (x_1\ x_2 \ldots x_n, x_{n+1})$ and $d' = (0\ 0 \ldots 0 1)$ are n + 1 vectors and A and c can be written in partitioned form as follows:

$$A = \begin{bmatrix} A_1 & 0 \\ A_2 & 0 \\ . & . \\ . & . \\ . & . \\ A_m & 0 \\ A_{m+1} & e \\ 0 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} C_1 \\ C_2 \\ . \\ . \\ . \\ C_m \\ C_{m+1} \\ M \end{bmatrix}$$

For $i = 1 \ldots m$. A_i is a matrix whose rows are formed by evaluating the vector $\left(\frac{\partial f_i}{\partial x_1}, \ldots \frac{\partial f_i}{\partial x_n}\right)$ at all grid points. Thus A_i is $N \times n$ where N is the number of grid points. Corresponding to A_i we have the N - vector C_i whose elements are given by

$$\sum_{j-1}^{n} \left.\frac{\partial f_i}{\partial x_j}\right]_{x^*} x_j^* - f_i(x^*)$$

evaluated at the grid points x^*.

Similarly, A_{m+1} is defined by the rows

$$\left(- \frac{\partial g}{\partial x_1} \quad \ldots - \frac{\partial g}{\partial x_n}\right)$$

and c_{m+1} is its corresponding vector with elements

$$g(x^*) - \sum_{j=1}^{n} \left.\frac{\partial g}{\partial x_j}\right]_{x^*} x_j^* \quad .$$

The $(n + 1)$ st column of A contains zeros in all rows except those corresponding to A_{m+1} and the last row which contains ones. Note that this last row corresponds to the restriction $x_{n+1} \leq M$.

The Dual Problem and Its Simplex Tableau

The dual to the linear problem described above is given by

minimize $c'w$

s.t.

$$A'w \geq d$$

$$w \geq 0$$

Here for convenience the restrictions have all been written as $A'w \geq d$ but in fact since x_{n+1} is unrestrained the last restriction is an equality.

This restriction has coefficients which are the last column of A and has a one on the right hand side.

This dual problem is best summarized in tableau form as shown below. This tableau gives the rules for constructing any desired column. Thus to generate a column of the type indicated by columns 1 through m+1 the following steps are necessary:

(a) Rows 1 through n are obtained by evaluating the indicated partial derivatives at the desired grid point.

(b) Row 0 is obtained by evaluating C_i as defined above at the desired grid point and changing its sign.

(c) Row n+1 is 0 for i=1 ... m and 1 for i=m+1.

	0	1	...	i	...	m	m+1	A	S_0	S_1	...	S_n
0	0			$-C_i^*$			$-C_{m+1}^*$	-M	1	0		0
1	0			$\partial f_i^* / \partial x_1$			$-\partial g^* / \partial x_1$	0	0	-1		.
.	.			.			.	.	.	0		.
.	.			.			.	.	.	.		.
.	.			.			.	.	.	.		0
n	0			$\partial f_i^* / \partial x_n$			$-\partial g^* / \partial x_n$	0	0	.		-1
n+1	1			0			1	1	0	0		0

TABLEAU FOR CONVEX PROGRAMMING

The Algorithm

Either the original or the dual linear problems formulated above can theoretically be solved by the simplex method but it is clear that for even moderately large problems and reasonable grid spacing the number of restrictions in the primal (or number of variables in the dual) would be so large as to exceed the capacity of most computers.

In this section a computational procedure is developed which solves the problem by a simplex-like algorithm which requires the formation of only a small number of the tangent planes described above.

Since the algorithm is an iterative process, it is only necessary to describe how to get started, how to proceed from one step to the next and when to stop. This will now be done using the standard terminology and notation from the simplex method.

At any iteration, say the k^{th}, a basis matrix call it B_k consisting of $n + 2$ columns from the tableau is required, or more precisely its inverse B_k^{-1} is required. To get started we have the initial basis matrix B_0 given in partitioned form by

$$B_0 = \begin{bmatrix} 1 & 0 & \cdots & -M \\ 0 & & -I_n & 0 \\ 0 & & 0 & 1 \end{bmatrix}$$

The inverse of B_0 is easily obtained by changing the sign on M.

Assume now that stage k of the iteration has been reached and B_k^{-1} is available. Denote the elements of B_k^{-1} by b^{ij}, $ij = 0, \ldots n + 1$. Recall that the first row of B_k^{-1} is the current pricing vector and that $b^{00} = 1$ hence the current pricing vector is $(1, b^{01}, \ldots b^{0n+1})$. Now assuming for the moment that all columns in the tableau have been formed, the simplex method requires the computation of the inner product of the pricing vector with a column of the tableau to see if this column is eligible to come into the basis. Looking first at columns of the type $i = 1 \ldots m$ this computation yields the net price

$$- C_i^* + \sum_{j=1}^{n} b^{0j} \frac{\partial f_i}{\partial x_j}\Big]_{x^*}$$

for a typical tangent plane at x^*. If this net price is ≥ 0 this vector is eligible to come into the basis. Since it is inconceivable to have all such vectors available for inspection the following question is raised: Are there any eligible tangent planes representing the i^{th} restriction and if so can the one with maximum net price be identified and constructed?

Looking at the above net price as a function of the unknown grid point and recalling the definitions of C_i^* we see that the net price for any tangent plane to $f_i(x)$ is given by

$$NP(x^*) = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\Big]_{x^*} (b^{0j} - x_j^*) + f_i(x^*)$$

Thus the net price is just the tangent plane formed from grid point x^* evaluated at $x_j = b^{0j}$. Now among all grid points, this ordinate is maximum for $x_j^* = b^{0j}$ and the magnitude of the net price is $f_i(b^{0j})$. Thus among all the vectors which can be formed from $f_i(x)$, the one with maximum net price is formed by allowing the grid point to have coordinates $x_j = b^{0j}$ and further, this maximum net price is given by evaluating f_i at this grid point. Thus if $f_i(b^{0j})$ is negative there are currently no columns in the i^{th} set which can come in and another set must be inspected.

Similarly, if the pricing vector is applied to the $(m + 1)$ st set of restrictions the net price is

$$- C_{m+1}^* - \sum_{j=1}^{n} \frac{\partial g}{\partial x_j}\Big]_{x^*} b^{0j} + b^{0\,n+1}$$

which upon rearrangement can be written as

$$NP(x^*) = b^{0\ n+1} - \left(\sum_{j=1}^{n} \left.\frac{\partial g}{\partial x_j}\right]_{x^*} (b^{0j} - x_j^*) + g(x^*) \right)$$

which is just $b^{0\ n+1}$ - (the ordinate at $x_j = b^{0j}$ to the tangent plane formed from grid point x_j^*). Again this net price is maximum for $x_j^* = b^{0j}$ and the maximum net price is given by $b^{0\ n+1} - g(b)$.

Thus in summary, the pricing operation consists simply of evaluating the functions $f_i(x)$, $i=1 \ldots m$ and $x_{n+1} - g(x)$ at $x_j = b^{0j}$, $j=1 \ldots n+1$. If any of these is positive then the corresponding vector can be brought in. In practice the one corresponding to the largest net price might be selected. The vector is then formed, up-dated and brought into the basis according to the usual simplex procedure to get B_{k+1}^{-1} and the process is repeated. The steps are repeated until all net prices are ≤ 0, or more reasonably, until no net price is $> e$ where e is some small positive number.

According to the dual theorem of linear programming the pricing vector for the terminal B^{-1} matrix gives the optimum solution. That is, $x_j = b^{0j}$ and $g(x) = b^{0\ n+1}$.

Observations

The sequence of pricing vectors generated as described above is, in fact, a sequence of trial solutions which converge on the true solution from outside the feasible region. Each trial solution is at the intersection of the n+1 tangent planes which are currently in the basis.

The choice of value of M requires some knowledge of the problem although the previous paragraph shows that if M is too small this will be indicated by the fact that $b^{0\ n+1} = M$ at termination. There is theoretically nothing wrong with choosing M too large.

Some problems have been encountered in which some of the hypotheses have not been satisfied in certain regions. For example, $\sqrt{x_1}$ is concave for $x_1 \geq 0$ but not real-valued for $x_1 < 0$. Thus if the pricing vector at some stage has $b^{01} < 0$ then the pricing operation can not be performed. However, in this case, the restriction $x_1 \geq 0$, that is vector S_1 in the tableau would be eligible to come in and if brought in, will remove the difficulty.

In general, if all assumptions are not satisfied it may be possible to apply the algorithm but care should be taken.

PART II: Documentation of the Program

I. Description of the Problem

As indicated in Part I the problem to be solved is:

$$\text{maximize} \quad g(x)$$
$$\text{subject to } f_i(x) \leq 0, \quad i = 1, \ldots, m ,$$

where $g(x)$ is concave and the $f_i(x)$ are convex, real valued functions of the n-vector x for all real x, and the functions are differentiable. Note that description as a minimization problem is also afforded, since minimizing $-g(x)$ is equivalent to maximizing $g(x)$.

For simplicity of programming we assume that the constraints f_i are segmented into four categories in the following order:

lower bound constraints denoted by $x_i \geq L_i$, $i = 1, \ldots,$ NLB

upper bound constraints denoted by $x_i \leq u_i$, $i = 1, \ldots,$ NUB

other linear constraints denoted by $Dx \leq b$, D is (NLIN) x (NVAR)

other convex constraints denoted by $h_i \leq 0$, $i = 1, \ldots,$ NCON .

Letting the functional be denoted by $h_{NCON + 1}$, the problem can be described as

$$\text{maximize} \quad h_{NCON + 1}(x)$$

subject to

$$-x_i + L_i \leq 0, \quad i = 1, \ldots, \text{NLB} \qquad (1)$$

$$x_i - U_i \leq 0, \quad i = 1, \ldots, \text{NUB} \qquad (2)$$

$$Dx - b \leq 0, \quad \text{where D is (N LIN x N VAR) and b is (NLIN x 1)} \qquad (3)$$

$$h_i(x) \leq 0, \quad i = 1, \ldots, \text{NCON}$$

II. Description of the Program

CONVEX is a subprogram written in FORTRAN IV (G) which executes the computational procedure consisting of simplex-like iterations with special pricing operations. The user is required to write a main program and function subprogram which, together, specify the particular problem CONVEX is to solve, and call CONVEX.

The main program will have available to CONVEX an array consisting of the lower and upper bounds and the left side of (3). This information on upper-and lower bounds will be provided through the single dimensioned arrays ILB, BLB, IUB, BUB, where

ILB	gives the subscripts of the variables having lower bounds
BLB	gives the lower bound for the ILBth variable
IUB	gives the subscripts of the variables having upper bounds
BUB	gives the upper bound for the IUBth variable.

For example, consider a seven-variable problem having upper and lower bounds as follows:

$$L_3 \leq x_3 \leq U_3 \qquad\qquad -x_3 + L_3 \leq 0$$

$$x_5 \leq U_5 \qquad \text{or, in proper input form,} \qquad -x_6 + L_6 \leq 0$$

$$L_6 \leq x_6 \qquad\qquad x_3 - U_3 \leq 0$$

$$x_5 - U_5 \leq 0$$

This information may be provided by having

ILB(1) = 3

ILB(2) = 6

BLB(1) = L_3

BLB(2) = L_6

IUB(1) = 3

IUB(2) = 5

BUB(1) = $-U_3$

BUB(2) = $-U_5$

The other linear constraints are entered as

$$A(I, J) = [D \ \vdots \ -b],$$

where $A\begin{pmatrix} x \\ \hdashline 1 \end{pmatrix} \leq 0$ represents constraints as in (3). Values of the following variables should also be supplied:

NVAR:	number of variables.
NET:	total number of constraints plus one.
NLB:	number of lower bound constraints.
NUB:	number of upper bound constraints.
NLIN:	number of other linear constraints.
NCON:	number of non-linear constraints.
BIGM:	upper bound on objective function.
DELTAM:	amount by which BIGM is increased if the original estimate of BIGM is too small.
NALTER:	If NALTER = 0, alternate net pricing is employed. Otherwise, alternate net pricing is employed except that priority restrictions are checked each iteration. If NALTER = 1, "priority" means lower bounds. If NALTER = 2, "priority" means lower and upper

bounds. If NALTER = 3, "priority" means lower and upper bounds and other linear constraints. If NALTER = 4, "priority" means all constraints.

EPS1: represents the stopping criterion associated with the net pricing operation, i.e., the program may stop if the maximum net price is less than EPS 1.

EPS2: represents the stopping criterion associated with the sum of the absolute value of changes in the values of the unknowns from iteration t to iteration t+k, i.e., $\sum_{i=1}^{NVAR} |X_i^t - X_i^{t+k}| \leq EPS2$.

NEPS2: represents k in the description of the stopping criterion associated with EPS2, and must be less than or equal to 15.

EPSPIV: represents pivot check stopping criterion, i.e., the program may stop if maximum element in entering vector is less than EPSPIV.

IBAS: instructs program concerning which basis to expect, and must be 0, 1, 2, or 3. For IBAS = 0, the initial basis inverse is BINV, supplied by the user. For IBAS = 1, the lower bound constraints are automatically used. For IBAS = 2, the upper bound constraints are automatically used. For IBAS = 3, the co-ordinates of an initial feasible point are supplied in X.

BINV: used only if IBAS = 0, in which case BINV contains the inverse of the initial basis and has dimension (NVAR + 1) x (NVAR + 1).

X: used only if IBAS = 0 or IBAS = 3, in which case X contains the current solution vector but has dimension NVAR.

DELTAX: used only if IBAS = 3, in which case it is the amount by which any artificial bound is incremented if it appears in a terminal solution.

INBAS: used only if IBAS = 0, in which case the user must supply in INBAS(I the index of the constraint represented in the Ith column of BINV.

INVERT: instructs program as to how many iterations to perform before re-inverting.

ITPRNT: tells program how many iterations are to be performed before intermediate results are printed.

JBIN: If JBIN = 0, B^{-1} is included in the information printed out. Otherwise, it is excluded.

In addition, the main program must include the following three statements:

```
DIMENSION BINV(60, 60), X(60), INBAS(60)
COMMON/BLK 1/A(60, 60), BLB(60), BUB(60), ILB(60), IUB(60)
COMMON/BLK 2/NVAR, NLB, NUB, NLIN, NCON, NET *
```

A function subprogram of the form SETUP(I, J, X, C) is required to provide h_i, i = 1, ..., NCON + 1, and $\frac{\partial}{\partial X_j} h_i$, i = 1, ..., NCON + 1; j = 1, NVAR. For specified I, SETUP will provide $h_i(x)|_{X*}$ when J = 0, and $\frac{\partial}{\partial X_j} h_i(x)|_{X*}$ j = 1, ..., NVAR when J = 1.

III. An Example

Consider the following problem:

$$\text{Maximize } 2X_1 - X_1^2 + X_2$$

$$\text{subject to } -.7X_1 + X_2 \leq 1$$

$$2X_1^2 + 3X_2^2 \leq 6$$

$$0 \leq X_2 \leq 1.3$$

$$X_1 \geq 0$$

*For the double-precision version, these arrays should be dimensioned 30, or 30 x 30, except for A, which should be 50 x 30.

In the format of Section I, the problem may be described as

$$\text{Maximize} \quad 2X_1 - X_1^2 + X_2$$

$$\text{subject to} \quad -X_1 + 0 \leq 0$$

$$-X_2 + 0 \leq 0$$

$$X_2 - 1.3 \leq 0$$

$$-.7X_1 + X_2 - 1 \leq 0$$

$$2X_1^2 + 3X_2^2 - 6 \leq 0$$

The input information described in Section 3 is as follows:

$$A = (-.7, 1, -1)$$

$$h_1(x) = 2X_1^2 + 3X_2^2 - 6$$

$$h_2(x) = 2X_1 - X_1^2 + X_2$$

$$\nabla_x h_1(x) = \begin{pmatrix} \frac{\partial h_1}{\partial X_1} \\ \frac{\partial h_2}{\partial X_2} \end{pmatrix} = \begin{pmatrix} 4X_1 \\ 6X_2 \end{pmatrix}$$

$$\nabla_x h_2(x) = \begin{pmatrix} \frac{\partial h_2}{\partial X_1} \\ \frac{\partial h_2}{\partial X_2} \end{pmatrix} = \begin{pmatrix} 2 - 2X_1 \\ 1 \end{pmatrix}$$

The main program may appear as follows:

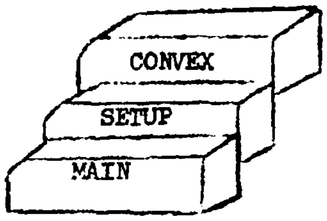
```
      DIMENSION BINV(60, 60), X(60), INBAS(60)
      COMMON/BLK 1/A(60,60), BLB(60), BUB(60), ILB(60), IUB(60)
      COMMON/BLK 2/NVAR, NLB, NUB, NLIN, NCON, NET
      NVAR = 2
      NET = 6
      NLB = 2
      NUB = 1
      NLIN = 1
      NCON = 1
      BIGM = 4.
      DELTAM = 1.
      NALTER = 0
      EPS1 = .001
      EPS2 = .001
      EPSPIV = .0001
      INVERT = 50
      NEPS2 = 10
      IBAS = 1
      ITPRNT = 1
      JBIN = 0
      ILB(1) = 1
      ILB(2) = 2
      BLB(1) = 0.
      BLB(2) = 0.
      IUB(1) = 2
      BUB(1) = -1.3
      A(1, 1) = -.7
      A(1, 2) = 1.
      A(1, 3) = -1.
      CALL CONVEX (IBAS, BINV, X, INBAS, NALTER, EPS1, EPS2, NEPS2, EPSPIV,
     $INVERT, ITPRNT, JBIN, BIGM, DELTAM, DELTAX)
      STOP
      END
```

SETUP may appear as follows:

```
      FUNCTION SETUP (I, J, X, COL)
      DIMENSION X(60), COL(60)
      SETUP = 0.
      IF (J. EQ. 0) GO TO (100, 200), I
      IF (J. EQ. 1) GO TO (300, 400), I
100   SETUP = 2.*X(1)**2 + 3.*X(2)**2 - 6.
      RETURN
200   SETUP = 2.*X(1) - X(1)**2 + X(2)
      RETURN
```

```
300     COL(1) = 4. * X(1)
        COL(2) = 6. *X(2)
        RETURN
400     COL(1) = 2. - 2. *X(1)
        COL(2) = 1.
        RETURN
        END
```

The program can then be assembled as,



REFERENCE

[1] Hartley, H. O., and R. R. Hocking, "Convex Programming by Tangential Approximation," Management Science, Vol. 9, No. 4, 1963.

List of Users of CONVEX

Mr. Harry Jones
Department of Civil Engineering
Texas A&M University
College Station, Texas 77843

Dr. Raymond Farrish
Department of Ag. Economics
University of Connecticut
Storrs. Connecticut 06268

Professor Dale Colyer
Department of Ag. Economics
University of Missouri
Columbia, Missouri 65201

Dr. J. D. Pegram
Department of Statistics
Mississippi State University
Starkville, Mississippi

Dr. Hollis H. Oxspring
Dept. of Quantitative Business Mgt.
University of Houston
Houston, Texas 77004

Dr. Larry Claypool
Dept. of Mathematics and Statistics
Oklahoma State University
Stillwater, Oklahoma

Mr. Jack E. Doyle
Department of Mathematics
Memphis State University
Memphis, Tennessee 38111

Mr. Walter Johnston
College of Ag. and Biological Sciences
Clemson University
Clemson, South Carolina 29631

Dr. Gary Carey
Department of Industrial Engineering
Texas A&M University
College Station, Texas 77843

Mr. Mike McKay
Institute of Statistics
Texas A&M University
College Station, Texas 77843

Mr. Roger C. Pfaffenberger
Institute of Statistics
Texas A&M University
College Station, Texas 77843

Mr. Richard Callen
Institute of Statistics
Texas A&M University
College Station, Texas 77843

Mr. Ernest Davis
Department of Ag. Economics
Texas A&M University
College Station, Texas 77843

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Texas A&M University	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP Unclassified

3. REPORT TITLE

CONVEX: A COMPUTER PROGRAM FOR SOLVING CONVEX PROGRAMS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Progress Report

5. AUTHOR(S) *(First name, middle initial, last name)*

H. O. Hartley, R. R. Hocking, L. R. LaMotte, and H. H. Oxspring

6. REPORT DATE July 1970	7a. TOTAL NO. OF PAGES 21	7b. NO. OF REFS 1

8a. CONTRACT OR GRANT NO. N00014-68-A-0140	9a. ORIGINATOR'S REPORT NUMBER(S) Number 23
b. PROJECT NO. NR047-700	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Office of Naval Research

13. ABSTRACT

~~This~~ The report describes a computer program implementing the Hartley-Hocking convex programming algorithm. The two parts of this report are, respectively, a description of the Hartley-Hocking method as extracted from the original paper, and the documentation of the computer program.

ATTACHMENT III

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

Unclassified
Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Convex						
Programming						
Optimization						

DD FORM 1 NOV 65 1473 (BACK)
(PAGE 2)

Unclassified
Security Classification